数理解析研究所講究録 1325 巻 2003 年 233-237

類推機能をもった 対話型シークェント計算証明システムの開発

山田 敬三, 平田 耕一, 原尾 政輝

九州工業大学 情報工学部
〒 820-8502 福岡県 飯塚市 川津 680-4
{yamada, hirata, harao}@ai.kyutech.ac.jp

あらまし: シークェント計算に基づく証明は人間にとって解かりやすく教育支援をはじめ多くの応用が可能である. しかしながら, 推論過程が複雑なため自動化は困難である. 本稿では, 類推機構を導入することによって半自動的に機能する対話型シークェント証明システムの開発をしたので, その概要について述べる.

Developping Intaractive Sequent Caliculus Prover Based on Analogy

Keizo Yamada, Kouich Hirata, and Masateru Harao

Department of Artificial Intelligence
Kyushu Institute of Tecnology
Kawazu 680-4, Iizuka 820-8502, Japan
{yamada, hirata, harao}@ai.kyutech.ac.jp

abstract: Since a proof based on sequent calculus is easy to understand for human, it has many applications including computer aided education. However, it is difficult to automate because its inference process is complex. We develop an interactive semi-automated sequent calculus prover by introducing an analogycal reasoning mechanism. In this paper, we demonstrate the outline of our system.

1 はじめに

現在, 自動証明システムの多くは, タブロー法や Prolog に代表される導出原理といった論理体系が用いられている. それらは, 計算機で処理しやすい反面, 証明過程は人間には理解しにくい. しかしながら, 論理学の教育支援に用いたり, 証明からプログラムを抽出する場合, あるいは数学の証明システムへの応用などでは, 証明可能か否かを問うだけではなく, その証明過程が重要になる. そこで, 人間が直観的に理解しやすい論理体系を自動化し証明過程の表示機能や操作機能などを計算機上に実現することで, 有用なシステムを構築することが期待できる. このような観点から, 我々は, 人間が直観的に証明過程を理解することのできる, 古典論理の体系である LK シークェント計算の自動化を行った.

LK シークェント計算には用いる規則が多数あり, 規則の適用の仕方が複雑になるため証明の自動化が困難である. そこで, 本稿ではヒューリスティックを用いた自動化の方法として, スキーマ誘導による類推 [1] を導入する. スキーマとは一般化された知識であり, 問題のパターンを表し, 類似性の判定に用いる. また, スキーマ誘導方式とは, スキーマと未知の問題との類似性から, その問題の解を導出する仕組みである. この類推の実現にあたっては, 多くのスキーマを蓄え, その中から適切なスキーマを探し出す必要がある. 本稿ではそのためのスキーマベースの設計と探索法について考察する.

LK 証明は木構造の図で表されるが, その証明図を計

算機上で見やすく表示するためには工夫が必要である. 証明図を表示するシステムとして, XIsabelle や xpe などがあるが, 証明の操作が簡潔でない, 述語論理が扱えないといった難点がある. このような難点を克服した, 直観的に見やすく操作性のよいシステムを構築すれば, 教育支援などへの応用が期待できる. 本稿では, まず, LK シークェント計算の証明図を直接表示できるシステムを作り, 類推によって自動的に得られた証明を表示したり, 規則を 1 ステップごとに簡潔な操作で適用して証明することもできる対話的証明機能を実現する.

本稿の構成は以下の通りである. まず, 2 節で, 本研究で用いる LK シークェント計算について述べ, 3 節で類推処理について述べる. 4 節では, 類推処理のためのスキーマベースの設計, および探索法について議論する. 最後に, 5 節でシステムの実装の概要について述べる.

2 準備

$A_1, \cdots, A_m, B_1, \cdots, B_n$ をそれぞれ論理式とする. このとき, 以下の 0 個以上の述語論理式の列の組を**シークェント** (*sequent*) という.

$$A_1, \cdots, A_m \vdash B_1, \cdots, B_n$$

シークェントは, 直観的には $A_1, \cdots, A_m$ を仮定すると $B_1, \cdots, B_n$ のいずれかが成り立つことを表している. さらに, $\vdash$ の左右に同じ論理式が現れるシークェントを**公理** (*axiom*) という. S_1, S_2, S をそれぞれシークェントとする. このとき, **推論規則** (*inference rule*) とは, 次の形をした図である:

$$\frac{S_1}{S}, \quad \text{または} \quad \frac{S_1 \quad S_2}{S}.$$

推論規則は, 直観的には $S_1 (S_2)$ が正しいとき, S は正しいことを表している. LK シークェント計算の推論規則にはいくつかの体系がある [3] が, 本稿では, 以下の 12 種類を用いる:

$$\frac{B, \Gamma \vdash \Delta \quad \Gamma \vdash A, \Delta}{A \supset B, \Gamma \vdash \Delta}(\supset\text{-左}) \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \supset B}(\supset\text{-右})$$

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}(\wedge\text{-左}) \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}(\vee\text{-右})$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}(\wedge\text{-右}) \qquad \frac{A, \Gamma \vdash \Delta \quad \Gamma \vdash \Delta, B}{A \vee B \Gamma \vdash \Delta}(\vee\text{-左})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p(z),\ q(z) \vdash p(z) \quad p(z),\ q(z) \vdash q(z)}{p(z),\ q(z) \vdash p(z) \wedge q(z)}\wedge\text{-右}}{p(z),\ \forall x.q(x) \vdash p(z) \wedge q(z)}\forall\text{-左}}{\forall x.p(x),\ \forall x.q(x) \vdash p(z) \wedge q(z)}\forall\text{-左}}{(\forall x.p(x)) \wedge (\forall x.q(x)) \vdash p(z) \wedge q(z)}\wedge\text{-左}}{(\forall x.p(x)) \wedge (\forall x.q(x)) \vdash \forall x.(p(x) \wedge q(x))}\forall\text{-右}}{\vdash (\forall x.p(x)) \wedge (\forall x.q(x)) \supset \forall x.(p(x) \wedge q(x))}\supset\text{-右}$$

図 1: $(\forall x.p(x)) \wedge (\forall x.q(x)) \supset \forall x.(p(x) \wedge q(x))$ の証明図

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}(\neg\text{-左}) \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}(\neg\text{-右})$$

$$\frac{\forall x.A,\ A[x := t],\ \Gamma \vdash \Delta}{\forall x.A,\ \Gamma \vdash \Delta}(\forall\text{-左})$$

$$\frac{\Gamma \vdash \Delta, \forall x.A,\ A[x := z]}{\Gamma \vdash \Delta, \forall x.A}(\forall\text{-右})$$

$$\frac{\exists x.A, A[x := z],\ \Gamma \vdash \Delta}{\exists x.A,\ \Gamma \vdash \Delta}(\exists\text{-左})$$

$$\frac{\Gamma \vdash \Delta, \exists x.A,\ A[x := t]}{\Gamma \vdash \Delta, \exists x.A}(\exists\text{-右})$$

ただし, 図中 $A[x := s]$ は A における変数 x の自由な出現を項 s で置き換えた式を表す. また, t は任意の項とし, z は変数条件を満たす変数とする. すなわち, z は下式に自由には現れない変数とする.

証明図とは, 最も下の式から出発して順次推論規則を適用して得られる図であり, シークェントを節とする木構造になっている. 論理式 P に対して, $\vdash P$ から始まる証明図を P の証明図といい, その葉がすべて公理となるとき, P は証明可能という. 例えば, 論理式 $(\forall x.p(x)) \wedge (\forall x.q(x)) \supset \forall x.(p(x) \wedge q(x))$ の証明図は図 1 のようになる.

3 スキーマ誘導による類推

本システムでは, スキーマ誘導型の類推 [1] を用いて証明を半自動化する.

スキーマとは論理式のパターンであり, 型付き二階項で定義される [4], 論理式に述語変数を許す表現である. 証明スキーマとはスキーマの証明である. スキーマとその証明スキーマの組を蓄積したものをスキーマベースという.

類推とは「似た問題は似た解法をもつ」という原理に基づく推論であり, 一般には正しいとは限らない. 類推証明においては, 推論の正しさを保存するような類似

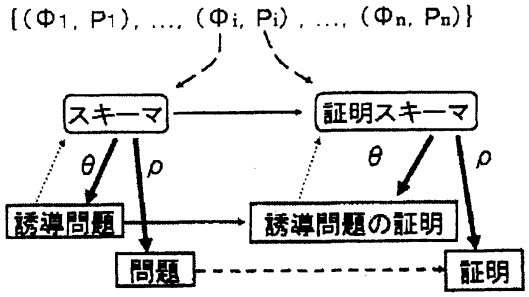

図 2: スキーマ誘導型類推の概要

性の定義が重要になる. そのため, 本システムでは, 類似性の判定にスキーママッチング [2] を用いる. スキーママッチングは, 論理的な正しさを保つ代入のみを導出するので, 提案するスキーマ誘導類推により常に正しい証明を得ることができる. 本システムにおける未知の問題 φ の類推による証明は以下の手順による:

まず, 証明の指針となるいくつかの誘導問題とその証明を抽象化し, スキーマベースに蓄える.

1. 未知の問題 φ とマッチする誘導問題のスキーマ Φ を探索する.
2. 探索に成功したとき,
   (a) $\Phi\theta = \varphi$ となる θ を計算する.
   (b) $Pr_\Phi\theta$ により φ の証明を導出する.
3. 探索に失敗したとき,
   (a) 手動で証明する.
   (b) その結果をスキーマベースに格納する.

4 スキーマベース

類推処理においてはマッチするスキーマを効率的に見つけることが重要となる. そのため, スキーマベースをスキーマの特徴に従って階層化し, 探索の効率化技法について述べる.

定義 1 スキーマ Φ の**特徴** (profile)C_Φ とは, Φ に出現するアトムをすべて新しい 0 引数の述語変数で置き換えた式である. 特徴 C_Φ に現れる論理結合子の数を C_Φ の大きさといい, $|C_\Phi|$ で表す.

直観的には, 特徴はスキーマの論理結合子に着目し, その配置を表す.

例えば, スキーマ $\Phi_1 = \forall x.P(x) \supset \exists x.P(x)$, $\Phi_2 = P \supset Q$, $\Phi_3 = P(a,b)$ の特徴はそれぞれ $C_{\Phi_1} = \forall x.P_1 \supset \exists x.P_2$, $C_{\Phi_2} = \neg P_3 \supset P_4$, $C_{\Phi_3} = P_5$ となり, その大きさは. $|C_{\Phi_1}| = 3, |C_{\Phi_2}| = 1, |C_{\Phi_3}| = 0$ と

$\Phi_1 = P \supset P,$
$\Phi_2 = (P \vee (Q \wedge \neg Q)) \supset P,$
$\Phi_3 = P \supset (\neg P \supset (P \wedge \neg P)),$
$\Phi_4 = P \supset (Q \supset P),$
$\Phi_5 = P \supset (Q \supset (P \wedge Q)),$
$\Phi_6 = P \supset (\neg P \supset Q),$
$\Phi_7 = (P \wedge Q) \supset P,$
$\Phi_8 = (P \wedge (P \supset Q)) \supset Q,$
$\Phi_9 = (\neg P \vee Q) \supset (P \supset Q),$
$\Phi_{10} = (P \supset R) \supset ((Q \supset R) \supset ((P \vee Q) \supset R)),$
$\Phi_{11} = \exists x.P(x) \vee \exists x.Q(x) \supset \exists x.(P(x) \vee Q(x)),$
$\Phi_{12} = \forall x(P(x) \supset Q(x)) \wedge \exists x.P(x) \supset \exists x.Q(x),$
$\Phi_{13} = \exists x.(P(x) \wedge Q(x)) \wedge \forall x(P(x) \supset h(x))$
$\quad \wedge (\forall x Q(x) \supset r(x)) \supset \exists x.(h(x)) \wedge r(x)),$
$\Phi_{14} = \forall x(P(x) \supset Q(x)) \wedge P(c) \supset \exists x.Q(x).$

図 3: スキーマベース SB に含まれるスキーマ

```
01: function search(φ, SB, C);
02:   while C ≠ ∅ do
03:     大きさが最大となる C ∈ C を適当に選ぶ;
04:     C := C − {C};
05:     if C と φ がマッチング可能 then
06:       while 各 (Φ, Pr_Φ) ∈ SB_C に対して do
07:         if Φ と φ がマッチング可能 then
08:           Pr_Φ とそのときのマッチング代入 θ を
                返して探索を終了する;
09:       end of while
10:       (φ は C とはマッチするが, SB_C 中の
              スキーマとはマッチしない)
11:       C := {C' ∈ C | C ≤ C'}
12:     end of if
13:   end of while
14:   探索に失敗して終了する
15: end of function
```

図 4: スキーマベース探索アルゴリズム

なる. 以降では, 特に混乱のない限り Φ の特徴を単に特徴という. また, ここで導入した述語変数の出現は高々1回となり, 引数も持たないので, 以降は特徴に現れる述語変数はすべて $*$ と表し, 束縛変数名も省略する. すると, 例えば, C_{Φ_1}, C_{Φ_2}, C_{Φ_3} は, それぞれ $\forall * \supset \exists *$, $\neg * \supset *$, $*$ と表わされる.

定義 2 特徴上の二項関係 $\leq$ と, 特徴 C に対するスキーマベース SB の部分集合 SB_C をそれぞれ次のように定義する:

$$\begin{aligned} \leq &= \{(C, D) \mid \text{ある代入}\theta\text{に対して}, C = D\theta\}, \\ SB_C &= \{(\Phi, Pr_\Phi) \in SB \mid \Phi \leq C\}. \end{aligned}$$

直観的には, SB_C は SB に蓄えられているスキーマのうち, 同じ特徴を持ったものの集合である.

命題 1 二項関係 $\leq$ に対して, 以下の2つの性質が成り立つ:

1. $\leq$ は半順序となる.
2. BS をスキーマベースとし, C_1, C_2 を特徴とする. このとき, $C_1 \leq C_2$ ならば, $SB_{C_1} \subset SB_{C_2}$ が成り立つ.

本システムでは, 大きさ3までの適当な特徴を用いてスキーマベースを階層化する. そして, 論理式 φ が与えられたとき, 特徴の集合 $\mathcal{C}$ を用いて階層化したスキーマベース SB 内を図4に示す探索アルゴリズム Search(φ, SB, $\mathcal{C}$) に従って探索する.

命題 2 スキーマベースを SB とし, φ を論理式とする. このとき, Search が φ とマッチング可能なスキーマの探索に失敗するとき, SB 内にそのようなスキーマは存在しない.

この性質はアルゴリズムの完全性を表している.

例 1 図4の14のスキーマを含むスキーマベース SB を以下の8の特徴を用いて図4のように階層化する:

$$\mathcal{C} = \left\{ \begin{array}{l} C_1 = * \supset *,\ C_2 = (* \vee *) \supset *, \\ C_3 = * \supset (* \supset *),\ C_4 = (* \wedge *) \supset *, \\ C_5 = (* \vee *) \supset \exists *, \\ C_6 = (* \vee *) \supset (* \supset *), \\ C_7 = (* \supset *) \supset (* \supset *), \\ C_8 = (* \wedge *) \supset \exists * \end{array} \right\}.$$

次に, 論理式 $\varphi = (\exists x.p(x) \wedge q) \supset \exists x.p(x)$ が与えられたときの SB の探索過程を示す:

1. まず, 図4中の特徴のうち, 最も大きい (最下段) ものの中から φ とマッチングするものを探す.

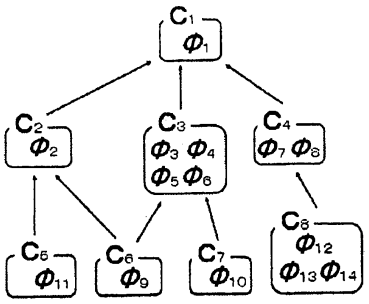

図 5: 階層化されたスキーマベース. ただし, $C_1, \cdots C_8$ は次の通り. $C_1 = * \supset *$, $C_2 = (* \vee *) \supset *$, $C_3 = * \supset (* \supset *)$, $C_4 = (* \wedge *) \supset *$, $C_5 = (* \vee *) \supset \exists *$, $C_6 = (* \vee *) \supset (* \supset *)$, $C_7 = (* \supset *) \supset (* \supset *)$, $C_8 = (* \wedge *) \supset \exists *$

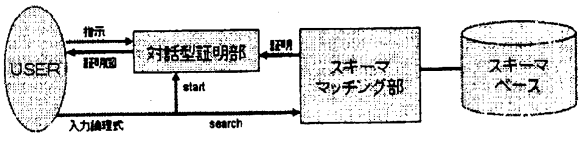

図 6: 本システムの概要

2. C_8 が見つかるので, $\Phi_{12}, \Phi_{13}, \Phi_{14} \in SB_{C_8}$ が φ とマッチング可能か否かを確める.
3. マッチングに失敗するので, $C_8 \leq C'$ となる C' のうち, もっとも大きなもは C_4 なので, SB_{C_4} について φ とのマッチング可能か否かを調べる.
4. 利用可能なスキーマとして $\Phi_7 \in C_2$ が見つかる.

5 実装

最後に本節では実装したシステムの使い方について述べる. 本システムでは, 類推による証明と1ステップごとに対話的に証明を進める対話型証明の2種類の方法が用意されている. まず, ユーザはシステムのテキストフィールドに証明したい論理式を入力する. ここで, "search" ボタンを押すと, システムはスキーマベースを探索し有効なスキーマをが見つかれば, それを援用して与えられた論理式の証明を返し, 見つからなければ, エラー処理をする.

論理式入力後, "start" ボタンを押すと, システムはその論理式のシークェントを表示して, 対話的証明を開始する. シークェント中の論理式は, すべてボタンで表されており, ユーザはそのボタンを押すことで, システムに, 推論規則を適用するシークェントと論理式を示す. システムは, その指示に従って適切な推論規則を

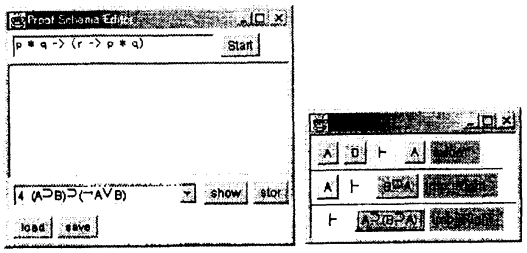

図 7: 論理式の入力 (左). スキーマベース内で見つかる有効なスキーマ (右).

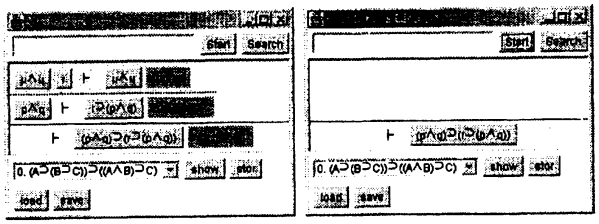

図 8: スキーマを援用して得られた証明 (左). スキーマベース内で見つかる有効なスキーマ (右)

シークェントに適用し, その結果を表示する. ただし, 公理に対してさらに推論規則を適用することはできないようになっている.

また, 証明図中の公理に含まれていない論理式ならば, どれでもユーザは選択し, 推論規則を適用することができる. このため, 間違えたときの証明のやり直しや, 別解を探すなど, 試行錯誤がしやすくなっている. この特徴は, 類推を用いて得られた証明に対しても同様である. システムの構成は図 5 のようになっている.

例えば, 論理式 $(p \wedge q) \supset (r \supset (p \wedge q))$ を類推を用いて証明する場合, 図 5 左のように入力し, "search" ボタンを押す. すると, システムは適当なスキーマ (図 5 右) を見つけ, それを基に証明を出力する (図 5 左). また, 論理式を入力した時点で, "start" ボタンを押せば, 図 5 右のように, 対話的に証明を進めるための準備がなされる.

6 考察

本稿では, 類推機能を持ったシークェント計算証明システムの概要について述べた. 特に, システムを構築する上での要所のひとつであるスキーマベースについて, その構成法と探索法を示た. スキーマの大きな特徴を捉えながら, 効率よく探索するアルゴリズム Search を定式化した. なお, Search には次の性質がある:

1. 探索アルゴリズムが, 出力したスキーマは正しい証明を導出する. 2. 利用可能なスキーマがあれば必ず探索は成功するという意味で完全である.

今後の課題として, 証明過程を再利用できるシステムの設計・構築が上げられる. そのためには以下の機能が必要となる:

1. その証明の自動的な抽象化.
2. スキーマベースの構成の自動化

また, 十分な証明を行うのに, 必要なスキーマを調べるのも今後の課題である.

参考文献

[1] Harao, M.: *Proof discovery in LK system by analogy* , LNCS 1345, pp.197–211, 1997.

[2] Kubo, K., Yamada, Y., Hirata K. and Harao, M.: *Efficient schema matching algorithm based on prechecking* , IEICE, J85-D-I, No 2, pp.143–151, 2002.

[3] Troelstra, A. S. and Schwichtenberg, H.: *Basec Proof Theory*, Cambridge University Press, 1996.

[4] Yamada, Y., Hirata K. and Harao, M.: *Schema matching and its complexity* , IEICE, J82-D-I, No 11, pp.1307–1316, 1999.